Reporte Grupal - Juego Flash Point Fire Rescue

Modelación de Sistemas Multiagentes con Gráficas Computacionales

Hecho por:

Diego Alfaro - A01709971

Rommel Toledo - A01709922

Introducción

Antecedentes

El juego del cual vamos a basar nuestra simulación se llama **Flash Point Fire Rescue**, este es un juego de mesa cooperativo en el que los jugadores asumen el papel de un bombero que tienen que rescatar en conjunto, un mínimo de siete víctimas antes de que el fuego se propague a través del edificio. Los jugadores deben moverse estratégicamente para extinguir llamas, romper muros, abrir puertas, revelar puntos de interés (potenciales víctimas) y mover víctimas a zonas seguras, todo esto con una cantidad limitada de acciones por turno y enfrentando el riesgo de que se derrumbe el edificio o sean noqueados por una explosión. Dado que cada turno introduce una nueva amenaza al avanzar el fuego, se tiene que jugar de manera cooperativa y planificada para poder conseguir la victoria.
El juego se escogió en sí debido a que presenta utilidad al momento de encontrar estrategias para resolver escenarios complejos como lo son los que ocurren en un desastre.

Dadas las características de este juego, es ideal como objetivo de un reto que busca simular un escenario en donde las capacidades de agentes diseñados puedan brillar.

Objetivos

Los principales objetivos que se tuvieron al llevar a cabo el reto fueron:

- Crear una simulación del juego Flash Point Fire Rescue mediante el uso de Mesa en Python usando agentes que tuvieran la función de bombero para intentar que lograran ganar el juego. Se busca

modelar correctamente todas las redes del juego así como acciones que pueden hacer los bomberos, esto para ser lo más fieles posibles a las reglas del modo de juego familiar.

- Crear la visualización de las simulaciones que sucedieran en Python dándole una representación visual a cada uno de los componentes de la simulación al igual que animando los eventos que suceden en el juego para permitir apreciar con facilidad lo que sucede en las simulaciones.

Alcance

Recursos:

- Equipo de 2 personas.
- Uso de Assets externos.

Entregables:

- Código documentado de Python con el modelo y agente, debe permitir la simulación del juego y estar apegado a las reglas del modo Familiar.
- Proyecto en Unity con código documentado, debe permitir visualizar correctamente los distintos *steps* que se dan a lo largo de la simulación en Python.

Cronograma:

- Primera Semana - Determinar la división de trabajo y los objetivos a cumplir.
- Segunda Semana - Empezar con el modelado de la simulación en Python, iniciando por crear el modelo.

- Tercera Semana - Creación del agente de la simulación de python, creación del código del servidor y corrección de errores y ajustes en el código de la simulación.
- Cuarta Semana - Obtención de Assets, diseño del código que permita el instanciamiento de los elementos del tablero, y conexión con el servidor.
- Quinta Semana - Creación de UI, ajuste de detalles finales y corrección de errores. Creación de los reportes grupales e individuales.

Metodología

Estrategia implementada

La estrategia utilizada fue dar un movimiento azaroso a los bomberos, es decir, permitiéndoles moverse en cualquiera de las direcciones posibles y válidas, pero con cierta tendencia a una priorización de acciones, sí se tiene una víctima buscarían sacarla de la casa, si no se tiene una víctima y hay fuego alrededor buscarían apagarlo y finalmente si no suceden los dos casos anteriores priorizaron revelar los puntos de interés.

Ventajas	Desventajas
• Al ser el movimiento mayormente aleatorio, el agente puede explorar su entorno. • Sirve como estrategia base para implementar estrategias más complejas.	• De igualmente al ser mayormente el aleatorio el movimiento causa que el agente no optimice sus puntos de acción. Por lo tanto tienen una eficiencia o rendimiento bajo. • Dada la influencia del azar, los resultados obtenidos suelen ser inconsistentes.

Esta estrategia facilita el diseño de los agentes al reducir la complejidad de su comportamiento, realmente no tiene mucho impacto en el diseño de la representación visual.

Esta estrategia fue determinada debido a las restricciones de tiempo y personal con las que contábamos nosotros.

Estructuras implementadas

Las estructuras de datos que se implementaron para la resolución de este proyecto fueron sets, diccionarios, y listas, viendo su mayor utilidad en:

- **Estructura de la grid:** Diccionario, donde la coordenada es usada como *key*, y tiene como valores listados las posiciones vecinas iniciando con la superior y yendo en sentido antihorario, al igual que un valor el cuál determinaba sí se tenía un muro (5), una puerta (2) o nada (1) entre la posición *key* y las posiciones adyacentes.
- **Posiciones del fuego y humo:** Para ambos se usaron sets, ya que permitían guardar coordenadas sin importar el orden y evitando la repetición de elementos.
- **Puntos de Interés (POI):** Diccionario donde se tiene como *key* la coordenada de la posición, y como valores dos booleanos que representaban, uno, sí se era una víctima o no, dos, sí el set ya se ha sido revelado por el bombero.
- **Vida de los muros:** Diccionario donde la *key* es la posición del muro y el valor su estado.

El uso de estás estructuras de datos sobre otras facilitó la implementación de la simulación debido a que eran muy útiles para representar y manejar los elementos que contenían y explicamos previamente.

Organización del grupo de trabajo

Dado que contamos con tiempo y personal limitado, ya que este reto fue diseñado para equipos de tres personas, nosotros decidimos segmentar el trabajo mediante especialización, uno de nosotros se haría mayormente cargo de la parte de la simulación en python al igual que el servidor y el otro se haría cargo de la parte de la visualización en unity. Ocasionalmente nos apoyaríamos entre nosotros al enfrentar complejidades.

Diseño

1. Establecer los objetivos del proyecto.
2. Determinar la división de trabajo.
3. Determinar la estrategia a implementar en base a lo que sería más sencillo de llevar a cabo.

Implementación

- Semana 2: Comenzar con el modelado de la simulación en Python, creando el modelo base que definiría las reglas del juego y las estructuras de datos necesarias.
- Semana 3: Desarrollar a los agentes para darles el comportamiento que tienen los bomberos en el juego, darle la estrategia planteada previamente y programar el servidor Flask para la comunicación entre Python y Unity. Corrección de errores.
- Semana 4: Obtener assets para la representación visual en Unity, empezar el diseño del código para instanciar elementos en la escena y establecer la conexión con el servidor.
- Semana 5: Desarrollar la interfaz de usuario (UI) en Unity, ajustar detalles finales y realizar pruebas de funcionalidad y corrección de errores en Python y Unity.

Operación

- Realizar pruebas a los agentes y modelo para evaluar si cumplen con las reglas del juego de una manera adecuada.
- Probar la integración entre Python y Unity para garantizar que los datos de la simulación se visualicen correctamente.
- Realizar ajustes finales, checar la documentación del código e intentar mejorar el rendimiento de la simulación.
- Crear los reportes grupales e individuales, documentando en base a la guía que se nos dió.

Resultados

Link del Vídeo: https://youtu.be/ruUJNVVeCTc